Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Nome completo do Autor

Título da monografia se for longo ocupa esta linha também

São Paulo
Dezembro de 2015

Título da monografia se for longo ocupa esta linha também

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Nome do Supervisor
[Cosupervisor: Prof. Dr. Nome do Cosupervisor]

São Paulo
Dezembro de 2015

Resumo

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave).

Palavras-chave: palavra-chave1, palavra-chave2, palavra-chave3.

Abstract

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa.

Keywords: keyword1, keyword2, keyword3.

Sumário

Capítulo 1

Introdução

Uma monografia deve ter um capítulo inicial que é a Introdução e um capítulo final que é a Conclusão. Entre esses dois capítulos poderá ter uma sequência de capítulos que descrevem o trabalho em detalhes. Após o capítulo de conclusão, poderá ter apêndices e ao final deverá ter as referências bibliográficas.

Para a escrita de textos em Ciência da Computação, o livro de Justin Zobel, *Writing for Computer Science* (Zobel, 2004) é uma leitura obrigatória. O livro *Metodologia de Pesquisa para Ciência da Computação* de Wazlawick (2009) também merece uma boa lida.

O uso desnecessário de termos em lingua estrangeira deve ser evitado. No entanto, quando isso for necessário, os termos devem aparecer *em itálico.*

```
Modos de citação:
indesejável: [AF83] introduziu o algoritmo ótimo.
indesejável: (Andrew e Foster, 1983) introduziram o algoritmo ótimo.
certo : Andrew e Foster introduziram o algoritmo ótimo [AF83].
certo : Andrew e Foster introduziram o algoritmo ótimo (Andrew e Foster, 1983).
certo : Andrew e Foster (1983) introduziram o algoritmo ótimo.
```

Uma prática recomendável na escrita de textos é descrever as legendas das figuras e tabelas em forma auto-contida: as legendas devem ser razoavelmente completas, de modo que o leitor possa entender a figura sem ler o texto onde a figura ou tabela é citada.

Apresentar os resultados de forma simples, clara e completa é uma tarefa que requer inspiração. Nesse sentido, o livro de Tufte (2001), *The Visual Display of Quantitative Information*, serve de ajuda na criação de figuras que permitam entender e interpretar dados/resultados de forma eficiente.

Capítulo 2

Desenvolvimentos

Embora neste exemplo tenhamos apenas um capítulo, entre a introdução e a conclusão de uma monografia podemos ter uma sequência de capítulos descrevendo o trabalho e os resultados. Estes podem descrever fundamentos, trabalhos relacionados, método/modelo/algoritmo proposto, experimentos realizados, resulatdos obtidos.

Cada capítulo pode ser organizado em seções, que por sua vez pode conter subseções.

Um exemplo de figura está na figura 2.1.

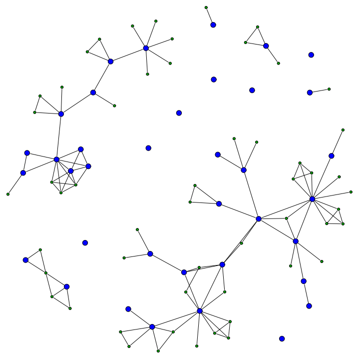


Figura 2.1: *Exemplo de uma figura.*

Capítulo 3

Conclusões

Texto texto[1].

[1]Exemplo de referência para página Web: www.vision.ime.usp.br/~jmena/stuff/tese-exemplo

Capítulo 4

Definições

Um grafo $G = (V, A)$ é um par ordenado, onde V e A são conjuntos disjuntos e cada elemento de A é um par não-ordenado de elementos de V. Os elementos de V são chamados de **vértices** e os de A são chamados de **arestas**.

Um grafo é dito **incremental** se suporta a adição de arestas. Analogamente, um grafo dinâmico é dito **decremental** se suporta a remoção de arestas. Um grafo incremental **ou** decremental é dito **parcialmente dinâmico**. Um grafo incremental **e** decremental é dito **totalmente dinâmico**.

Capítulo 5

Florestas dinâmicas

O problema da floresta dinâmica consiste em manter uma coleção de árvores enraizadas disjuntas que são modificadas ao longo do tempo pela adição e remoção de arestas. De forma objetiva, queremos realizar às seguintes operações:

- inicializa(n): Cria n árvores disjuntas, todas com apenas um vértice. Os vértices são numerados de 1 até n.
- raiz(v): Devolve a raiz da árvore que contém v.
- liga(u, v): Combina as árvores contendo u e v pela adição da aresta $\{u, v\}$. Supõe que u e v estão em árvores diferentes.
- corta(u, v): remove a aresta $\{u, v\}$. Supõe que a aresta $\{u, v\}$ existe.

Ao longo deste capítulo, denotamos por n o parâmetro da operação inicializa, isto é, o número de vértices da floresta. Também denotamos por m o número de operações raiz, liga e corta.

Uma maneira de resolver o problema é armazenar a floresta numa matriz de adjacência. Com essa representação, cada operação liga e corta consome tempo $O(1)$, mas a operação raiz consome tempo proporcional a profundidade do vértice, que é $O(n)$.

Ao representar as árvores de outra forma, reduziremos a complexidade da operação raiz para $O(\log n)$, pagando o preço de aumentar o consumo de tempo das operações de liga e corta também para $O(\log n)$.

5.1 Sequências

Uma **string** é uma sequência finita de elementos.

5.2 Sequências de Euler

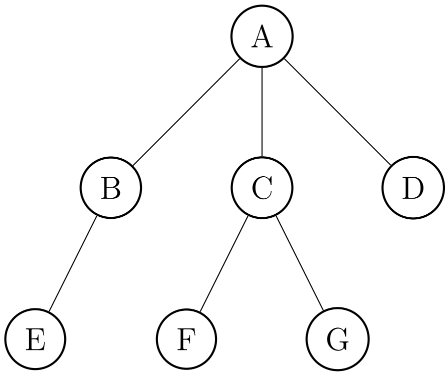
Definimos a **sequência de Euler** de uma árvore enraizada num vértice r por ET(r), onde ET é definido como:

function ET(u)
  acrescente uu
  for all v vizinho de u **do**
    acrescente uv
    if v não foi visitado **then**

ET(v)
end if
end for
end function

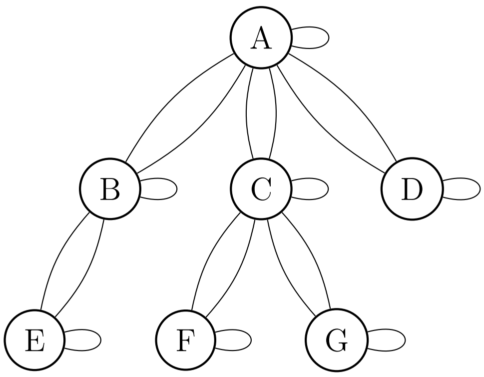
Ao aplicar o procedimento no vértice `A` da figura 5.1, vemos que a sequência eureliana é `AA AB BB BE EE EB BA AC CC CF FF FC CG GG GC CA AD DD DA`. Note que a raiz da árvore sempre será o primeiro elemento da sequência.

Figura 5.1: *Árvore enraizada no vértice A*



A sequência eureliana de uma árvore é na verdade o circuito eureliano do grafo gerado pela duplicação de suas arestas e inclusão de laços em todos seus vértices (veja a figura 5.2).

Figura 5.2: *Grafo induzido pela duplicação e adição de laços na árvore da figura 5.1*



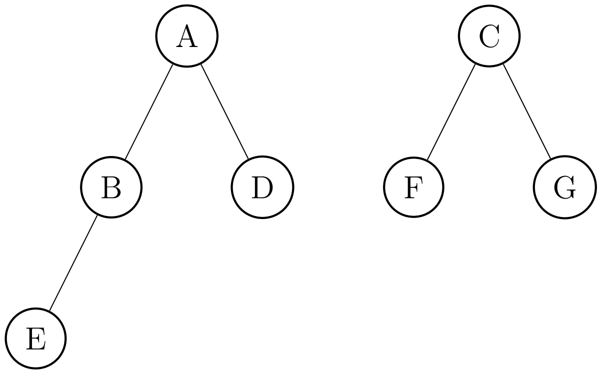
Note que uma sequência eureliana define unicamente uma árvore, já que contém todos seus vértices e arestas. Por esse motivo, podemos representar árvores por sequências eurelianas. Essa representação é enxuta e nos permite realizar as operações de liga, corta e raiz de forma simples, como veremos adiante.

Proposição 5.3. *Uma sequência eureliana de uma árvore com n vértices tem tamanho $3n - 2$.*

Demonstração. Seja T uma árvore com n vértices. Cada aresta de T aparece duas vezes na sequência e cada vértice aparece uma vez na forma de um laço, portanto temos $2(n-1) + n = 3n - 2$ elementos. □

Vamos analisar o que acontece com a sequência eureliana da árvore da figura 5.1 quando removemos a aresta AC. A figura 5.4 ilustra o resultado da operação.

Figura 5.4: *Resultado da remoção da aresta AC.*



Se tomarmos o vértice C como raiz da nova árvore que contém C, a sequência eureliana dessa árvore é (5.4) `CC CF FF FC CG GG GC`. Mantendo o vértice A como raiz da outra árvore, temos a sequência eureliana `AA AB BB BE EE EB BA AD DD DA`. Nesse caso, vemos que a sequência eureliana da árvore que contém C é uma subsequência contínua da sequência eureliana da árvore antes da remoção da aresta AC. Similarmente, a sequência eureliana da árvore que contém A é uma concatenação de duas subsequências contínuas da sequência eureliana original.

$$\begin{array}{l}\rightarrow \texttt{AA AB BB BE EE EB BA } \underline{\texttt{AC}} \texttt{ CC CF FF FC CG GG GC } \underline{\texttt{CA}} \texttt{ AD DD DA} \\ \rightarrow \texttt{AA AB BB BE EE EB BA} \qquad \texttt{CC CF FF FC CG GG GC} \qquad \texttt{AD DD DA}\end{array} \tag{5.4}$$

Proposição 5.5. *Sejam T uma árvore e S uma sequência eureliana de T. Para toda aresta e de T, é possível obter uma sequência eureliana das duas árvores de $T-e$ com quatro cortes e uma concatenação .*

Demonstração. Seja $e = uv$ uma aresta de T. Suponha sem perda de generalidade que `uv` ocorre antes de `vu` em S. Logo, S é da forma S_1 `uv` S_v `vu` S_2, onde S_v é a sequência eureliana da árvore de $T-e$ enraizada em v. Com quatro cortes extraímos S_v e removemos `uv` e `vu`. Uma concatenação S_1S_2 nos dá a sequência eureliana da nova árvore enraizada em u. □

Continuando com nosso exemplo, vamos analisar o que acontece quando adicionamos a aresta DG na floresta da figura 5.4. O resultado da operação é ilustrado na figura 5.6.

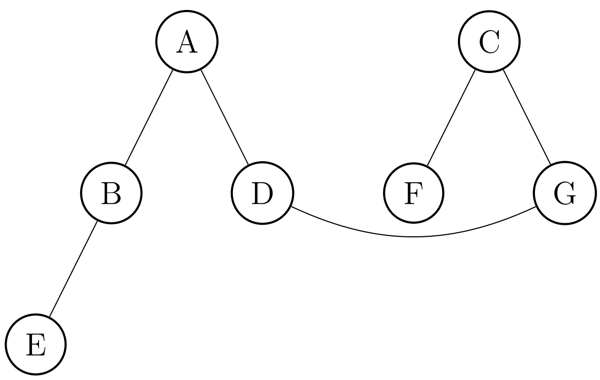
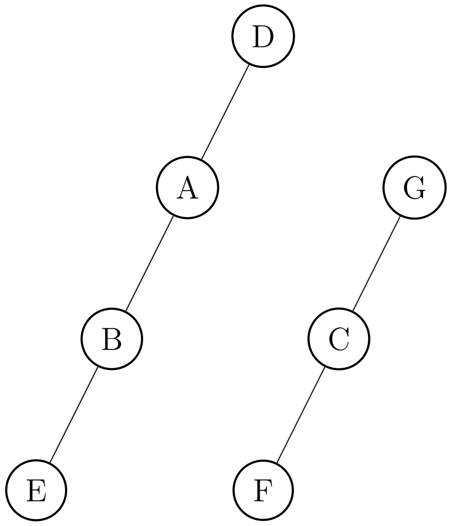


Figura 5.6: *Resultado da adição da aresta DG.*

Há algo estranho na figura. Ainda que ela represente corretamente a nova árvore, vamos dar um passo atrás e re-enraizar as árvores nos vértices D e G, como ilustrado na figura 5.7.

Figura 5.7: *Árvores re-enraizadas nos vértices D e G.*



Note que ao re-enraizar uma árvore, sua sequência eureliana muda. Nesse caso, as novas sequências são `DD DA AA AB BB BE EE EB BA AD` e `GG GC CC CF FF FC CG`. Ambas são rotações das sequências originais. Ao se adicionar a aresta DG nessa nova configuração, a nova sequência é `DD DA AA AB BB BE EE EB BA AD DG GG GC CC CF FF FC CG GD`, uma simples concatenação das sequências originais e da nova aresta.

Proposição 5.8. *Sejam T_1 e T_2 duas árvores disjuntas e S e R sequências eurelianas de T_1 e T_2, respectivamente. Para todo vértice $u \in T_1$ e $v \in T_2$, é possível obter a sequência eureliana de $(T_1 \cup T_2) + uv$ com quatro concatenações e um corte .*

Demonstração. Sejam u e v vértices de T_1 e T_2, respectivamente. Note que $S = S_1\,\mathtt{uu}\,S_2$ e $R = R_1\,\mathtt{vv}\,R_2$. Definimos $S' := \mathtt{uu}\,S_2S_1$ e $R' := \mathtt{vv}\,R_2R_1$. Logo, $S'\,\mathtt{uv}\,R'\,\mathtt{vu}$ é uma sequência eureliana de $(T_1 \cup T_2) + uv$ enraizada em u. □

As proposições apresentadas reduzem o problema da floresta dinâmica para o problema de concatenar e fatiar sequências. No capítulo 6 apresentamos uma maneira eficiente de se representar e realizar as operações em sequência.

Capítulo 6

Representação de sequências

Como visto no capítulo 5, desejamos uma representação eficiente de sequências que suporte as operações de concatenação e fatiamento. Uma solução simples é representar uma sequência com uma lista ligada e realizar as operações em $O(n)$, onde n é o tamanho da maior sequência. Entretanto, apresentamos uma maneira que utiliza árvores de busca binária balanceadas, reduzindo o tempo das operações para $O(\log n)$.

6.1 Árvores de busca binária balanceadas implícitas

Numa árvore de busca binária balanceada, ou ABBB, todo elemento possui uma chave. Ao se tratar de uma sequência, as chaves dos elementos serão as posições dos elementos na sequência, que serão armazenadas de forma implícita. Cada nó da árvore guarda, ao invés de uma chave, o tamanho de sua subárvore.

Uma treap, também conhecida como árvore cartesiana, é uma árvore de busca binária aleatorizada. Tem implementação relativamente simples quando comparada à árvores rubros-negras ou AVLs. Duas operações primitivas em treaps se mostram suficientes para realizar a maioria das operações.

- divide(T, chave)
- concatena(T_1, T_2)

Cada nó de uma treap guarda uma *chave* e uma *prioridade*. As chaves respeitam a estrutura de uma árvore de busca binária, enquanto as prioridades respeitam a estrutura de um *max-heap*.

Apêndice A

Título do apêndice

Texto texto.

Referências Bibliográficas

Tufte(2001) Edward Tufte. *The Visual Display of Quantitative Information*. Graphics Pr, 2nd edição. Citado na pág. 1

Wazlawick(2009) Raul S. Wazlawick. *Metodologia de Pesquisa em Ciencia da Computação*. Campus, primeira edição. Citado na pág. 1

Zobel(2004) Justin Zobel. *Writing for Computer Science: The art of effective communication*. Springer, segunda edição. Citado na pág. 1